UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

GENERATING REAL-WORLD AND SYNTHETIC GRAPH DATASET FOR GNN APPLICATION

Arpandeep Khatua

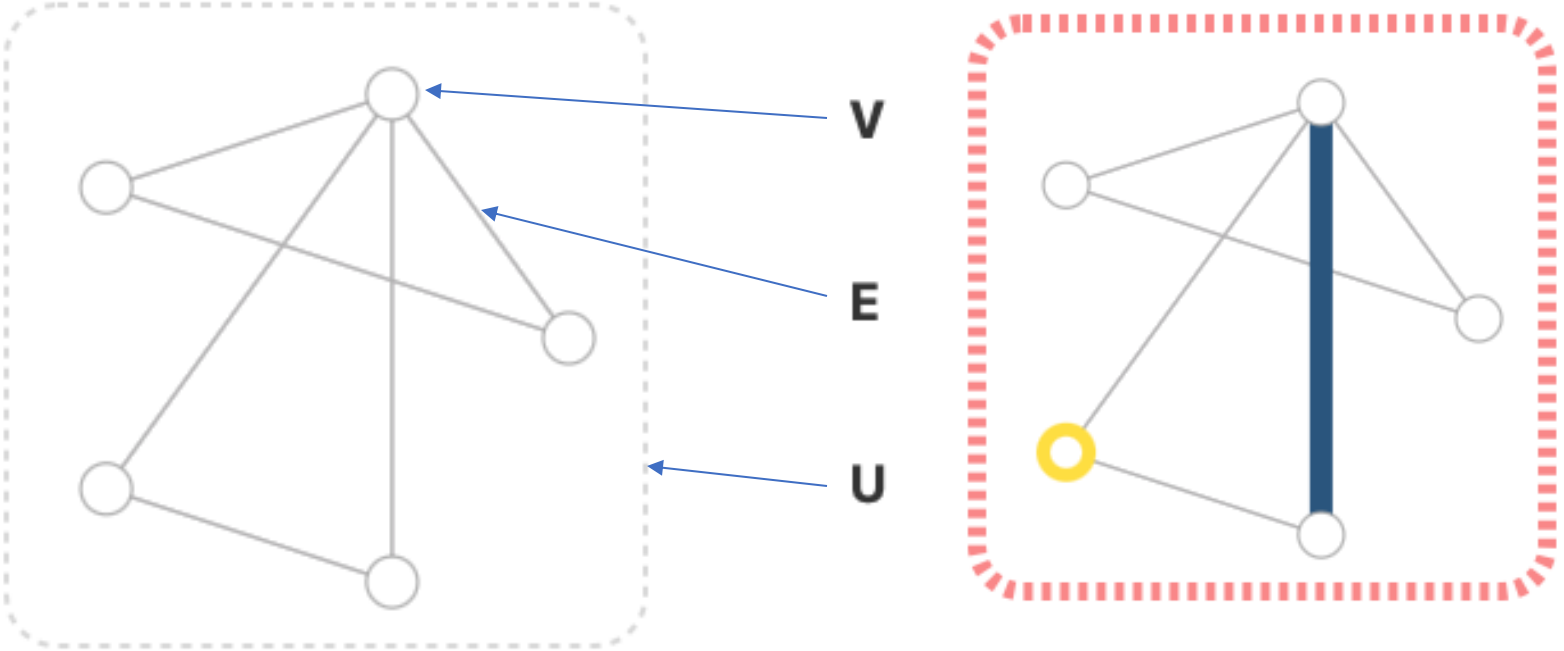
Adviser: Professor Wen-Mei Hwu

Mentor: Vikram Sharma Mailthody

20th April 2022

The IMPACT Research Group

WHAT ARE GRAPHS AND WHY DO WE NEED THEM?



Vertex (or node) embedding

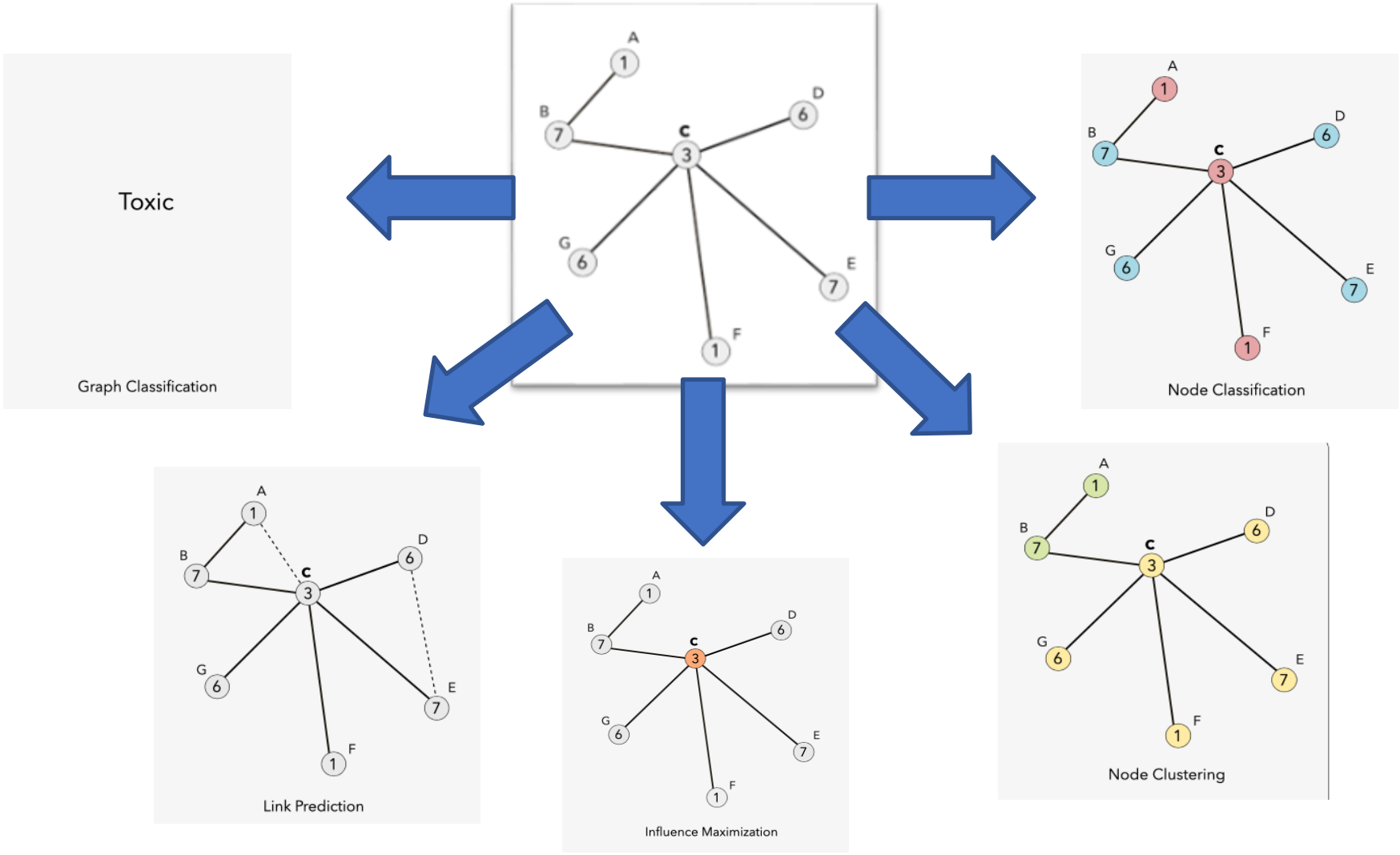
Edge (or link) attributes and embedding

Global (or master node) embedding

Sanchez-Lengeling, et al., "A Gentle Introduction to Graph Neural Networks", Distill, 2021.

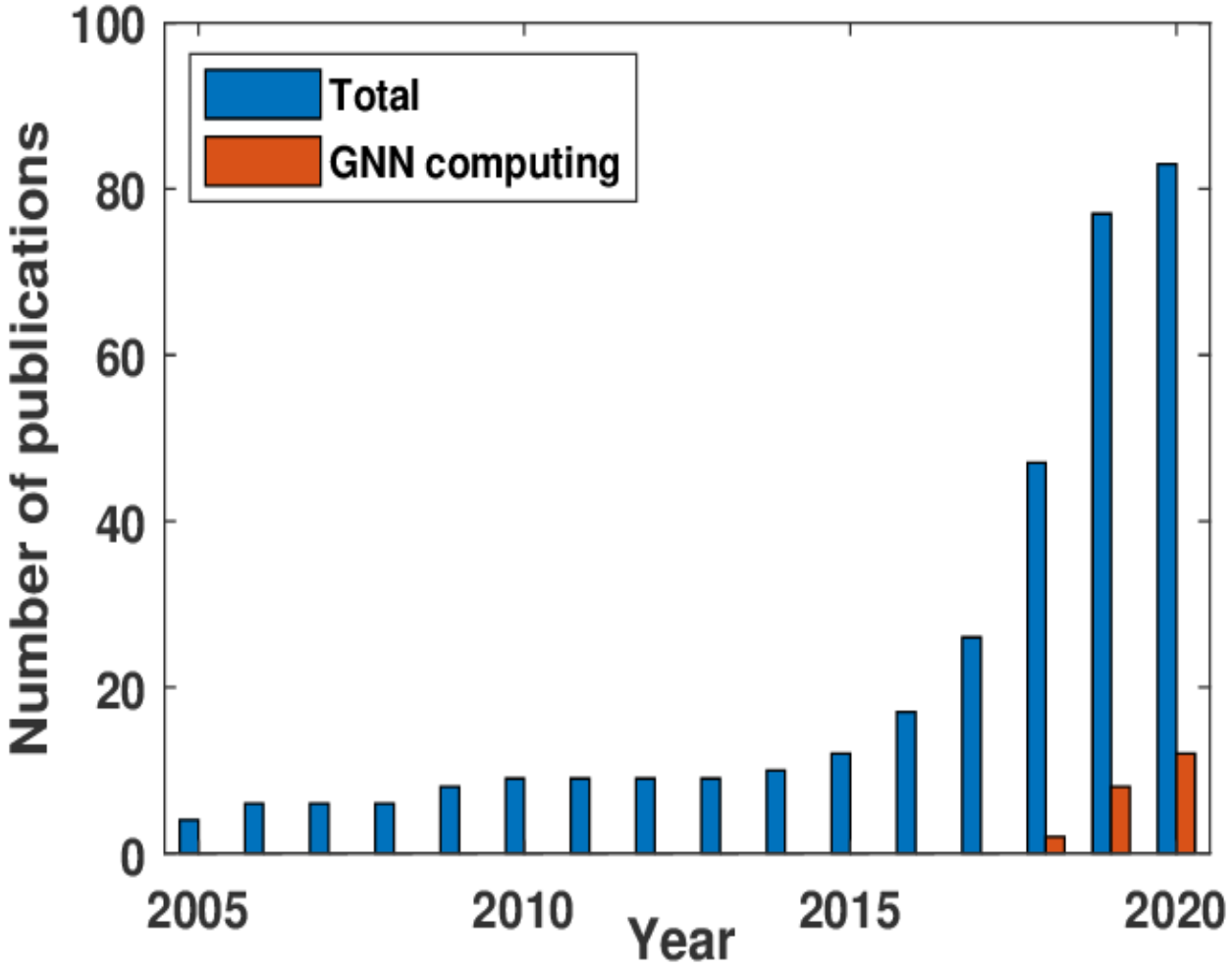
OKAY... SO WHAT CAN WE DO WITH GRAPHS?

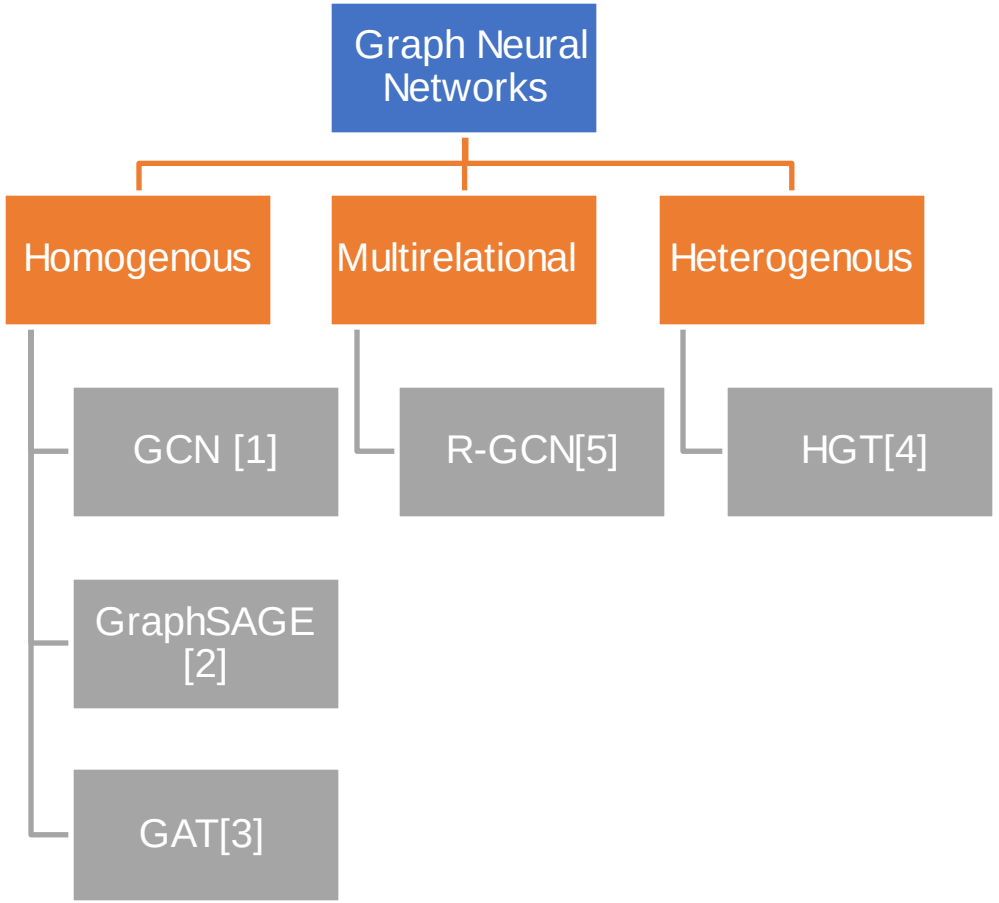
There are three general types of prediction tasks on graphs: graph-level, node-level, and edge-level.



Daigavane, et al., "Understanding Convolutions on Graphs", Distill, 2021.

DEEP LEARNING IN GRAPHS





https://arxiv.org/pdf/2010.00130.pdf

MAIN COMPONENTS OF MY RESEARCH WORK

As a part of the thesis, I had two main tasks:

1. To understand the latest GNN models and their limitations.

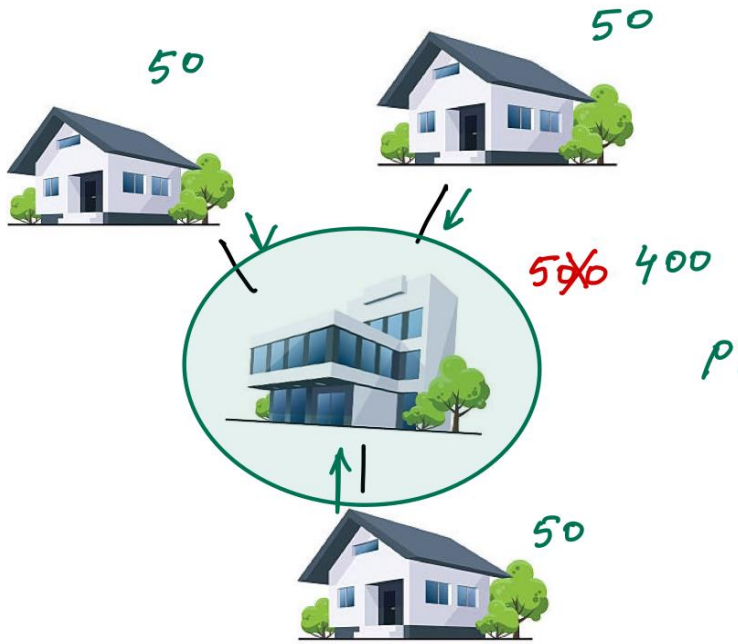
2. Generating a large graph dataset for both systems designer and large graph datasets.

Our final goal is to create a large graph dataset. However, this is not possible without understanding how GNN Models and previous graph datasets were made and work.

DEEP LEARNING IN GRAPHS

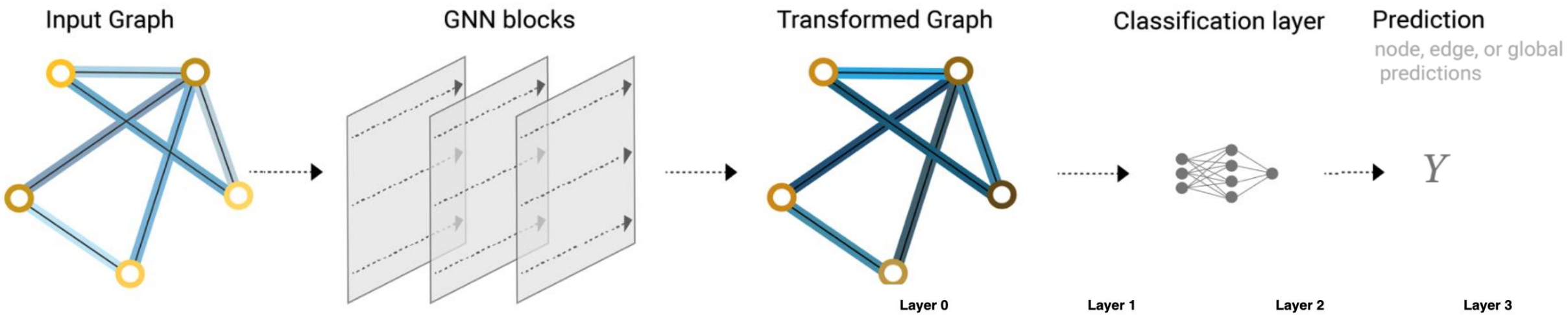
Basic House $\xrightarrow{Emb}$ e $\qquad P(e) = 50$

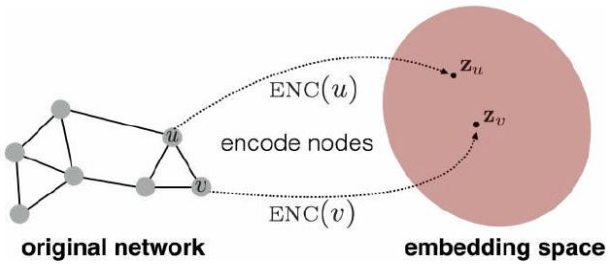
Mansion $\xrightarrow{Emb}$ X $\qquad P(X) = 500$

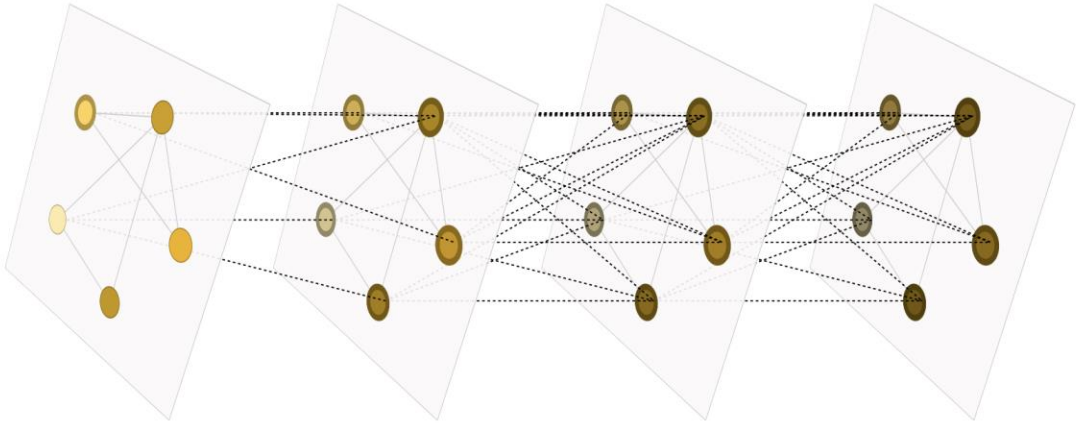


Graph neural network (GNN) - 101

Graph neural networks (GNNs) are a family of neural networks that can operate naturally on graph-structured data.







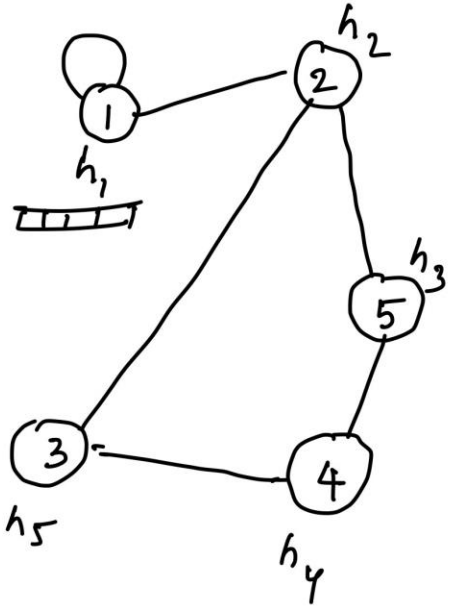
Sanchez-Lengeling, et al., "A Gentle Introduction to Graph Neural Networks", Distill, 2021.

LET'S TAKE A DEEP DIVE INTO THE WORKING OF A SIMPLE GNN

$$h_i' = \sigma\left(\sum_{j \in N(i)} W * h_j\right)$$

LET'S TAKE A DEEP DIVE INTO THE WORKING OF A SIMPLE GNN

$$h_i' = \sigma\left(\sum_{j \in N(i)} W * h_j\right)$$



① $\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$ * $\begin{matrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{matrix}$ * $\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$ = h_1'

learnable weight

Adj Matrix

[5, 5]

(sparse)

[5, 4]

[4, x]

dense

LET'S TAKE A DEEP DIVE INTO THE WORKING OF A SIMPLE GNN

$$h_i' = \sigma\left(\sum_{j \in N(i)} W * h_j\right)$$

$$h_i^{l+1} = \sigma\left(\sum_{j \in N_i} \frac{1}{c_{ij}} W^l * h_j^l\right)$$

No normalizing term → After a few layers this will cause the node embeddings to explode.

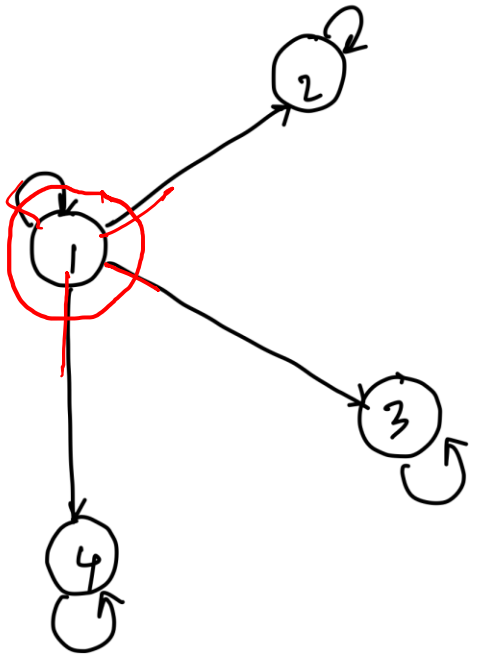
Add a normalizing term based on the degree of the node.

https://arxiv.org/pdf/1609.02907.pdf

GRAPH CONVOLUTIONAL NEURAL NETWORK

$$h_i^{l+1} = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} W^l * h_j^l\right)$$

$$H^{(l+1)} = \sigma\left(\underbrace{\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}}_{\text{Norm}} \underbrace{H^{(l)} W^{(l)}}_{\text{projection}}\right)$$



$[N, N]$

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot \end{bmatrix}$$

$A + I$

$$\tilde{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$[N, N]$

Degree

$\tilde{D}$

$[N, N]$

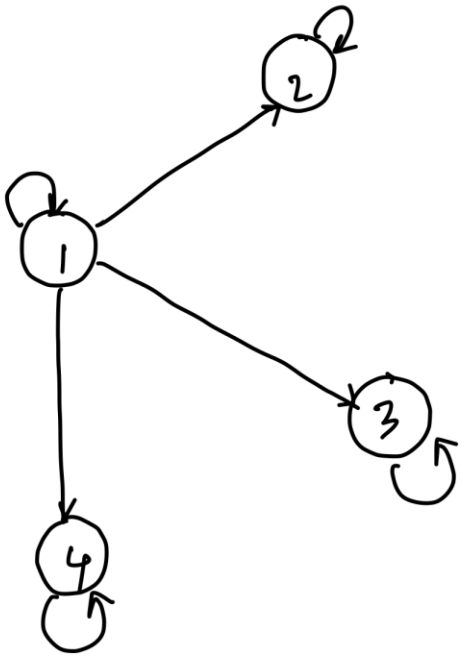
$$\tilde{D} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

https://arxiv.org/pdf/1609.02907.pdf

GRAPH CONVOLUTIONAL NEURAL NETWORK

$$h_i^{l+1} = \sigma\left(\sum_{j \in N_i} \frac{1}{c_{ij}} W^l * h_j^l\right)$$

$$H^{(l+1)} = \sigma\left(\underbrace{\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}}_{\text{Norm}} \underbrace{H^{(l)} W^{(l)}}_{\text{projection}}\right)$$



$\tilde{A}$ = A + I

$$\tilde{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$[N, N]$

a_{12}

Degree

$$\tilde{D} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$[N, N]$

$\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} \rightarrow a_{ij}$ ← Normalization

$$\frac{a_{ij}}{\sqrt{d_i d_j}} \quad \text{eg} \rightarrow \quad \frac{a_{12}}{\sqrt{d_1 d_2}}$$

$$\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} = \begin{bmatrix} \frac{a_{ij}}{\sqrt{d_i d_j}} & \cdots \\ \vdots & \ddots \end{bmatrix}$$

$[N, N]$

https://arxiv.org/pdf/1609.02907.pdf

LET'S TAKE A DEEP DIVE INTO THE WORKING OF A SIMPLE GNN

$$h_i' = \sigma\left(\sum_{j \in N(i)} W * h_j\right)$$

$$h_i^{l+1} = \sigma\left(\sum_{j \in N_i} \frac{1}{c_{ij}} W^l * h_j^l\right)$$

$$h_i^{l+1} = \sigma\left(W \cdot \left[\underset{j \in N_i}{\text{AGGREGATOR}}(\{h_j^l\}), h_i^l\right]\right)$$

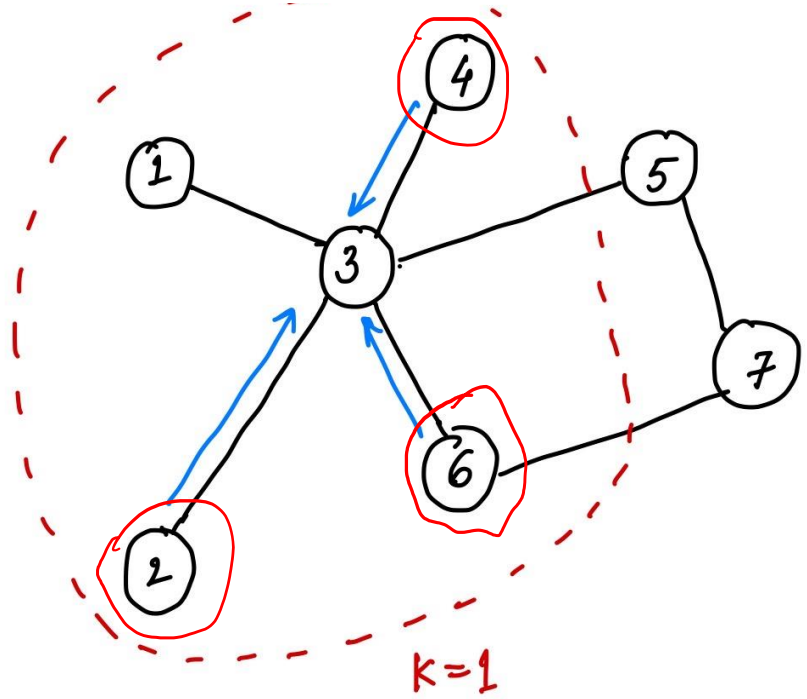
Update to a node embedding in GCN requires looking at all nodes. Very expensive operation

GraphSAGE: add sample and aggregator layer.

https://arxiv.org/pdf/1706.02216.pdf

GRAPHSAGE NETWORK

$$h_i^{l+1} = \sigma\left(W \cdot \left[\underset{j \in \mathcal{N}_i}{\text{AGGREGATOR}}\left(\{ h_j^{l} \} \right), h_i^{l} \right] \right)$$



Types of aggregator

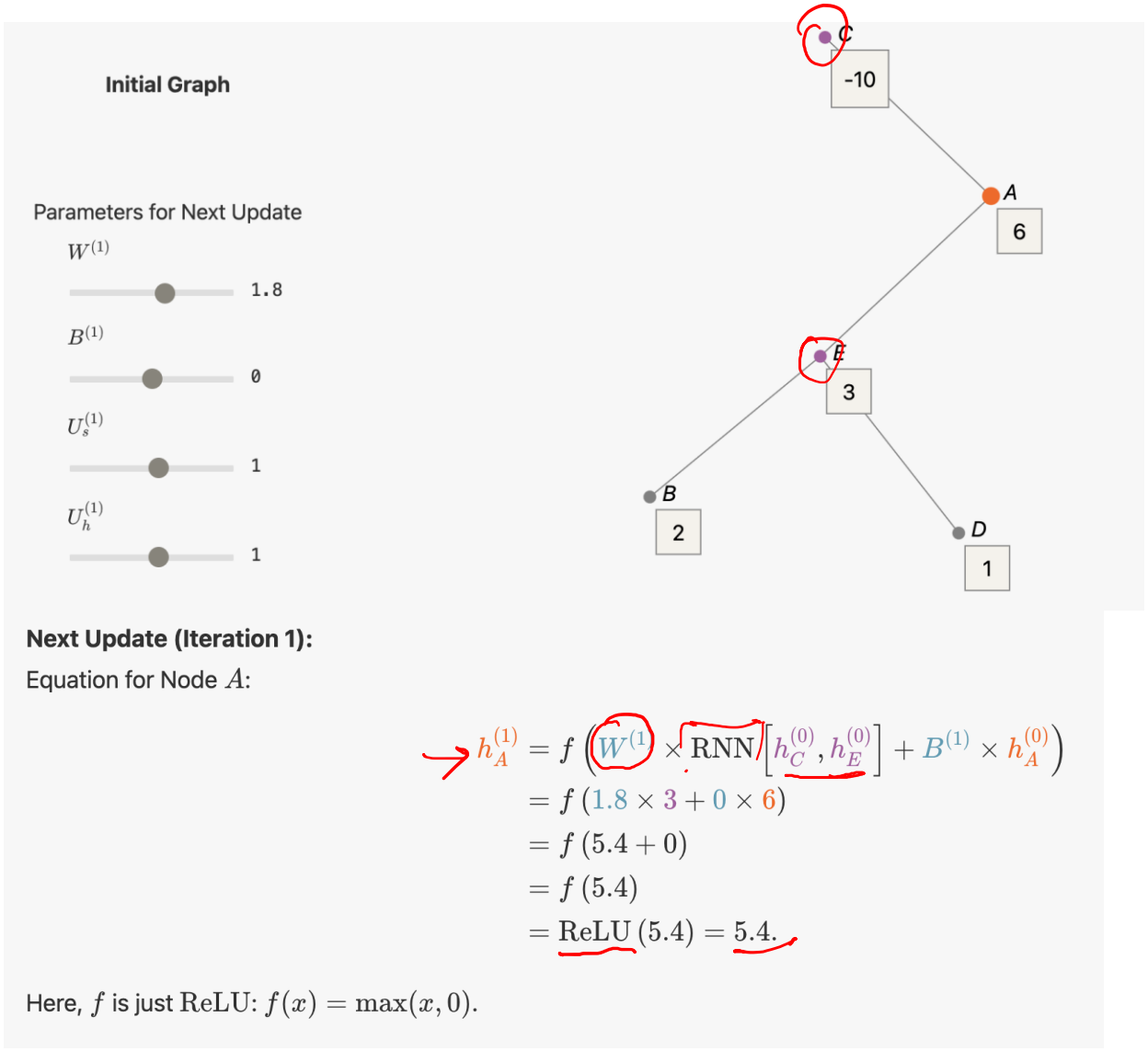
MEAN : $\underset{j \in \mathcal{N}_i}{\text{MEAN}}\left(\{ h_i^{l} \} \cup \{ h_j^{l} \} \right)$

POOLING : $\max\left(\{ \sigma\left(W_{pool} h_j^{l} + b \right), \forall j \in \mathcal{N}_i \} \right)$

And many more types of aggregators ...

https://arxiv.org/pdf/1706.02216.pdf

GRAPHSAGE NETWORK

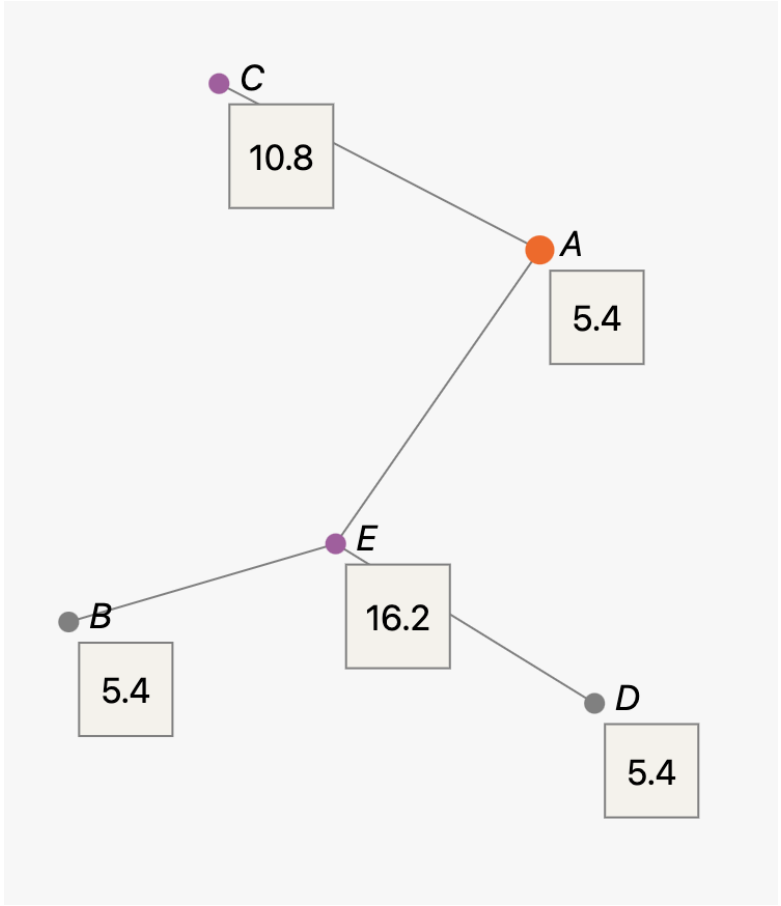


Next Update (Iteration 1):

Equation for Node A:

$$
\begin{aligned}
h_A^{(1)} &= f\left(W^{(1)} \times \mathrm{RNN}\left[h_C^{(0)}, h_E^{(0)}\right] + B^{(1)} \times h_A^{(0)}\right) \\
&= f\left(1.8 \times 3 + 0 \times 6\right) \\
&= f\left(5.4 + 0\right) \\
&= f\left(5.4\right) \\
&= \mathrm{ReLU}\left(5.4\right) = 5.4.
\end{aligned}
$$

Here, f is just ReLU: $f(x) = \max(x, 0)$.

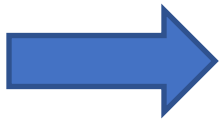


https://arxiv.org/pdf/1706.02216.pdf

LET'S TAKE A DEEP DIVE INTO THE WORKING OF A SIMPLE GNN

$$h_i' = \sigma\left(\sum_{j \in N(i)} W * h_j\right) \qquad h_i^{l+1} = \sigma\left(\sum_{j \in N_i} \frac{1}{c_{ij}} W^l * h_j^l\right)$$

$$h_i^{l+1} = \sigma\left(W \cdot \left[\underset{j \in N_i}{\text{AGGREGATOR}}\left(\{h_j^l\}\right), h_i^l\right]\right)$$



$$\vec{h_i}' = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_{ij}^k W^k \vec{h_j}\right)$$

Random sampling does not provide **attention** to important nodes.

GAT: Add attention layer with learned parameter to weight the neighbors.

https://arxiv.org/pdf/1710.10903.pdf

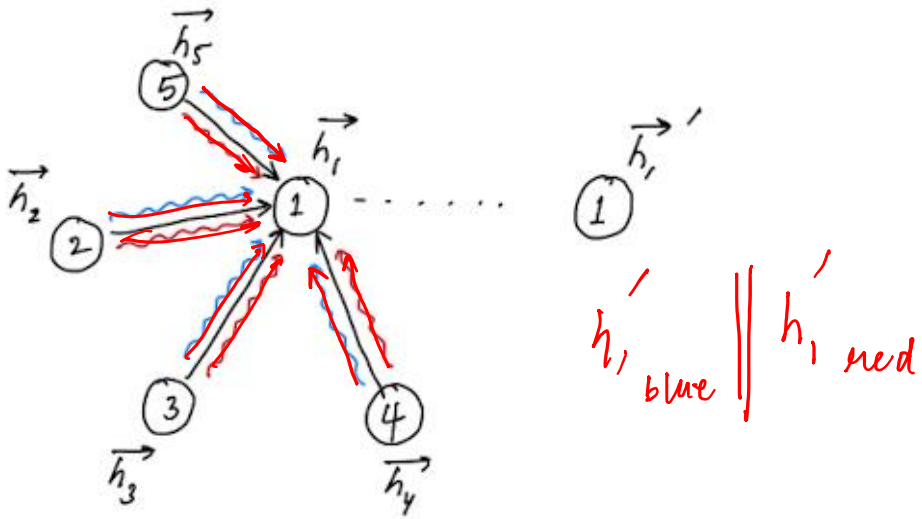
GRAPH ATTENTION NETWORK

Single head :

$$\vec{h_i}' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W \vec{h_j}\right)$$

Multi-headed :

$$\vec{h_i}' = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{k} W^{k} \vec{h_j}\right)$$



$\vec{h_1}'_{\text{blue}} \Vert \vec{h_1}'_{\text{red}}$

$\alpha_{ij} = \text{Softmax}(e_{ij})$

↑ Normalized attention coef.

→ Attention coefficients :

$e_{ij} = a(W\vec{h_i}, W\vec{h_j})$

↑ attention mechanism

https://arxiv.org/pdf/1710.10903.pdf

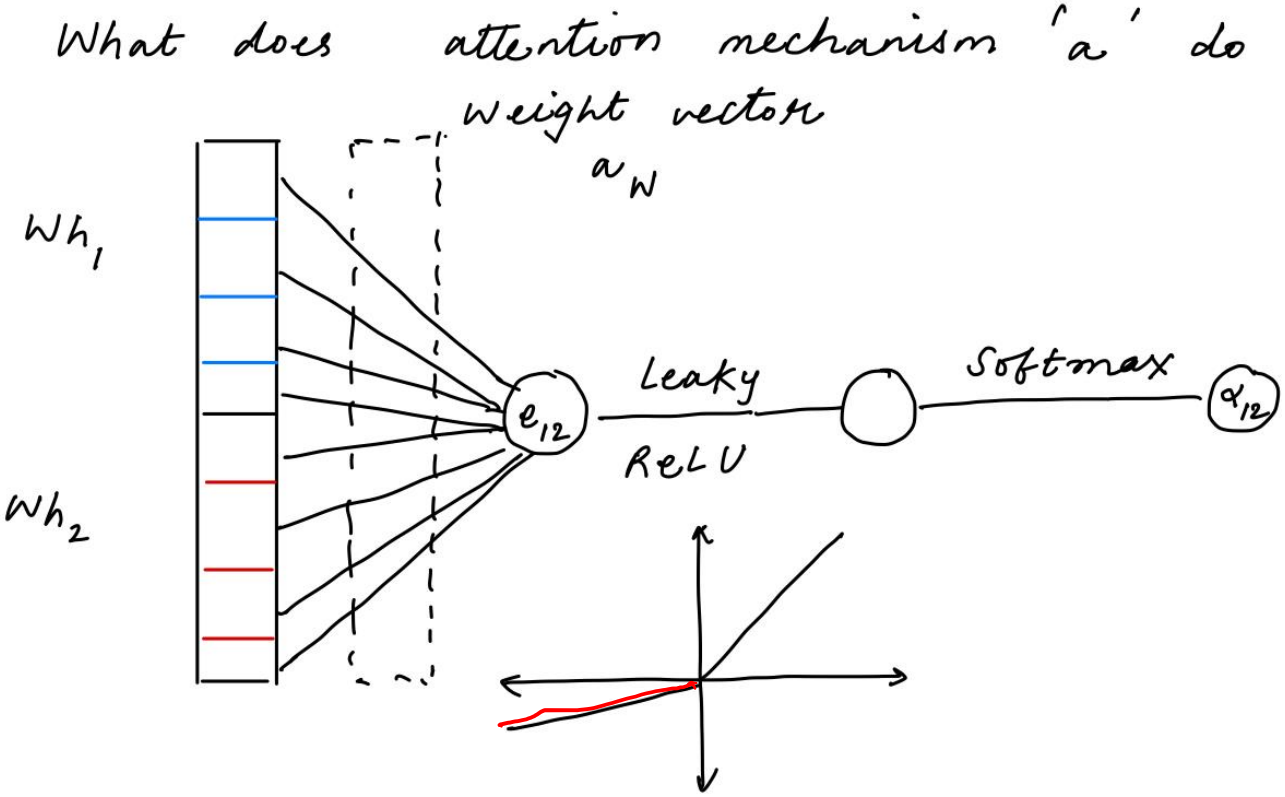
GRAPH ATTENTION NETWORK

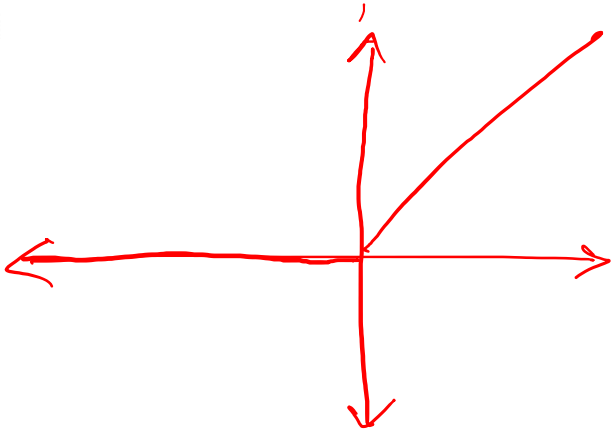
att. mechanism

$e_{ij} = a(Wh_i, Wh_j)$ attention coeff.

$\alpha_{ij} = \text{softmax}(e_{ij})$ normalized attention coeff.

What does attention mechanism 'a' do ?





https://arxiv.org/pdf/1710.10903.pdf

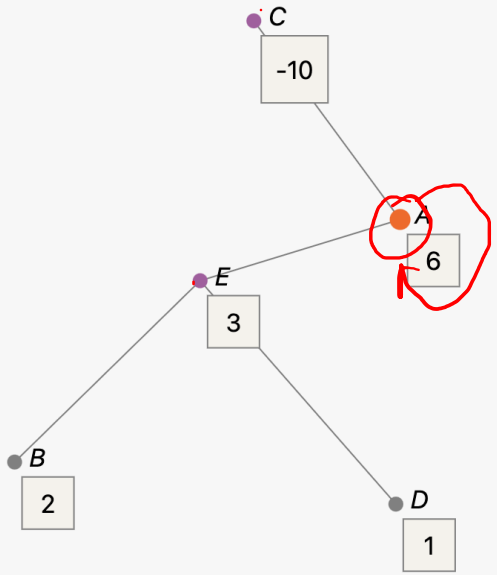
GRAPH ATTENTION NETWORK

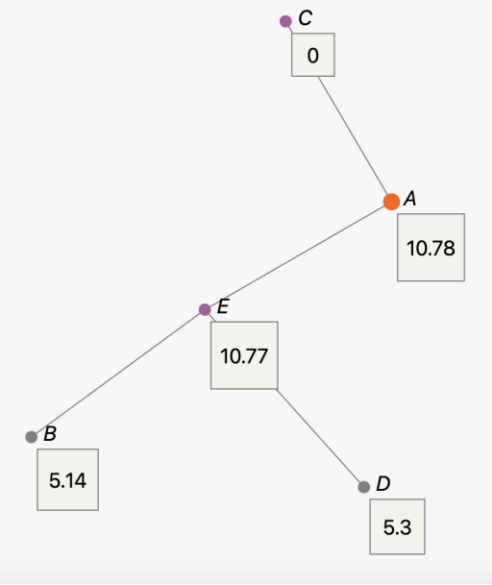
Initial Graph

Parameters for Next Update

$W^{(1)}$ 1.8

$A_W^{(1)}$ 1





Next Update (Iteration 1):

Equation for Node A:

$$
\begin{aligned}
h_A^{(1)} &= f\left(W^{(1)} \times \left(\alpha_{AC} \times h_C^{(0)} + \alpha_{AE} \times h_E^{(0)} + \alpha_{AA} \times h_A^{(0)}\right)\right) \\
&= f\left(W^{(1)} \times \left(0 \times h_C^{(0)} + 0 \times h_E^{(0)} + 1 \times h_A^{(0)}\right)\right) \\
&= f\left(W^{(1)} \times \left(0 \times -10 + 0 \times 3 + 1 \times 6\right)\right) \\
&= f\left(W^{(1)} \times \left(0 + 0.01 + 5.97\right)\right) \\
&= f\left(1.8 \times (5.99)\right) \\
&= f\left(10.78\right) \\
&= \mathrm{ReLU}\left(10.78\right) = 10.78.
\end{aligned}
$$

with attention weights α_A computed as:

$$
\begin{aligned}
\alpha_{AC} &= \frac{\exp\left(e_{AC}\right)}{\exp\left(e_{AC}\right) + \exp\left(e_{AE}\right) + \exp\left(e_{AA}\right)} \approx 0. \\
\alpha_{AE} &= \frac{\exp\left(e_{AE}\right)}{\exp\left(e_{AC}\right) + \exp\left(e_{AE}\right) + \exp\left(e_{AA}\right)} \approx 0. \\
\alpha_{AA} &= \frac{\exp\left(e_{AA}\right)}{\exp\left(e_{AC}\right) + \exp\left(e_{AE}\right) + \exp\left(e_{AA}\right)} \approx 1.
\end{aligned}
$$

where the unnormalized attention weights e_A are given by:

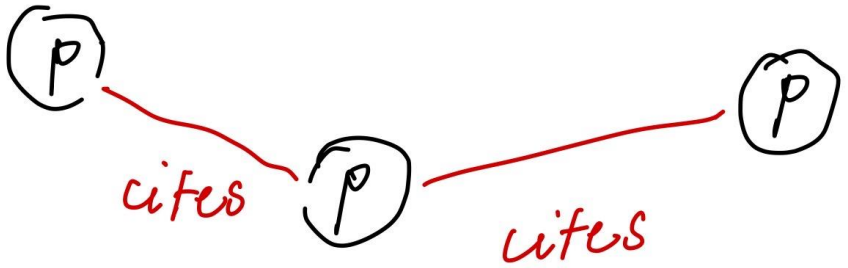
$$
\begin{aligned}
e_{AC} &= \mathrm{ReLU}\left(A_W^{(1)}\left(W^{(1)} h_A^{(0)} + W^{(1)} h_C^{(0)}\right)\right) = \mathrm{ReLU}\left(1\left(1.8 \times 6 + 1.8 \times -10\right)\right) = \mathrm{ReLU}\left(-7.2\right) = 0. \\
e_{AE} &= \mathrm{ReLU}\left(A_W^{(1)}\left(W^{(1)} h_A^{(0)} + W^{(1)} h_E^{(0)}\right)\right) = \mathrm{ReLU}\left(1\left(1.8 \times 6 + 1.8 \times 3\right)\right) = \mathrm{ReLU}\left(16.2\right) = 16.2. \\
e_{AA} &= \mathrm{ReLU}\left(A_W^{(1)}\left(W^{(1)} h_A^{(0)} + W^{(1)} h_A^{(0)}\right)\right) = \mathrm{ReLU}\left(1\left(1.8 \times 6 + 1.8 \times 6\right)\right) = \mathrm{ReLU}\left(21.6\right) = 21.6.
\end{aligned}
$$

We have omitted the superscripts on the attention weights for clarity.

Here, f is just ReLU: $f(x) = \max(x, 0)$.

https://arxiv.org/pdf/1710.10903.pdf

TYPES OF GRAPHS

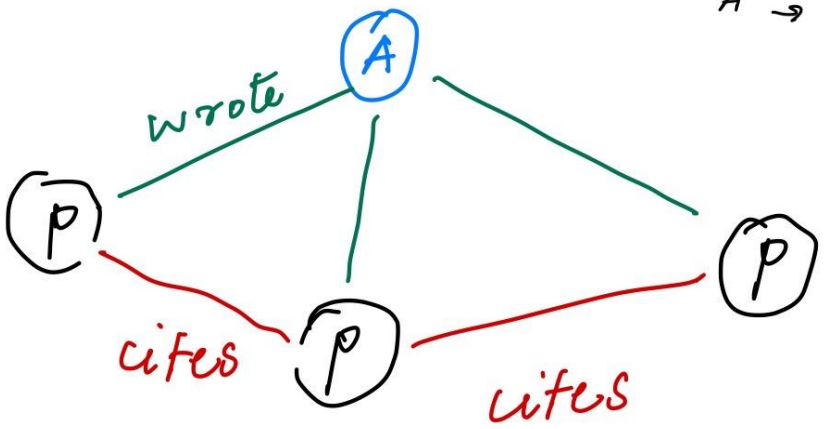


Apply GCN, GraphSAGE, GAT

(or many other models)

Homogeneous graph → 1 type of node, 1 type of relation

TYPES OF GRAPHS



P → Paper Node

A → Author Node

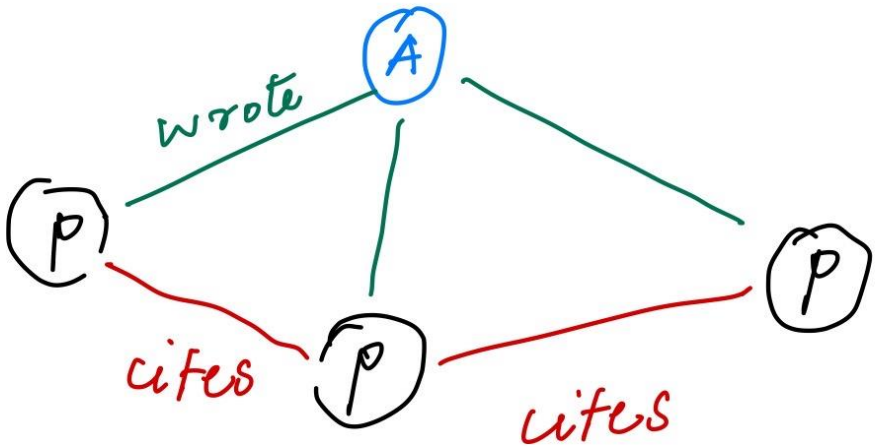
What models to apply?

① Simplify

② Heterogeneous Graph Models
eg: HAN, HGT

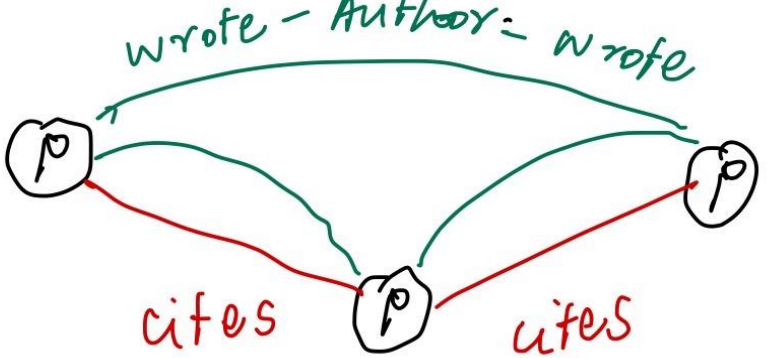
Heterogeneous graph → Multiple types of nodes,
Multiple types of relations.

TYPES OF GRAPHS



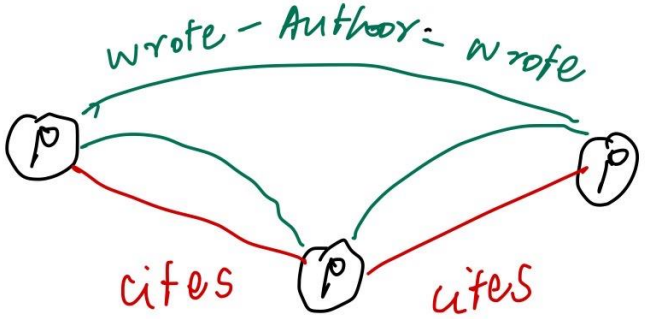
Heterogeneous graph

metapaths



Multirelational Graphs

TYPES OF GRAPHS



Apply Relational GCN / GAT

Multirelational Graphs → 1 type of node
Multiple types of relations.

MOVING FROM HOMOGENEOUS MODELS TO RELATIONAL MODELS

$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}(i)} W * h_j\right) \qquad h_i^{l+1} = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} W^l * h_j^l\right) \qquad h_i^{l+1} = \sigma\left(W \cdot \left[\underset{j \in \mathcal{N}_i}{\text{AGGREGATOR}}(\{h_j^l\}), h_i^l\right]\right) \qquad \vec{h_i}' = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h_j}\right)$$

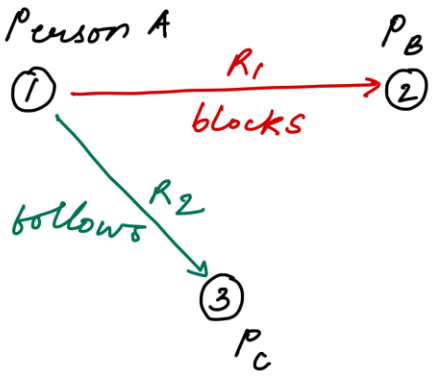
$$h_i^{l+1} = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} W^l * h_j^l\right) \quad \Rightarrow \quad h_i^{l+1} = \sigma\left(W_0^l h_i^l + \sum_{r \in R} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{ij}} W_r^l h_j^l\right)$$

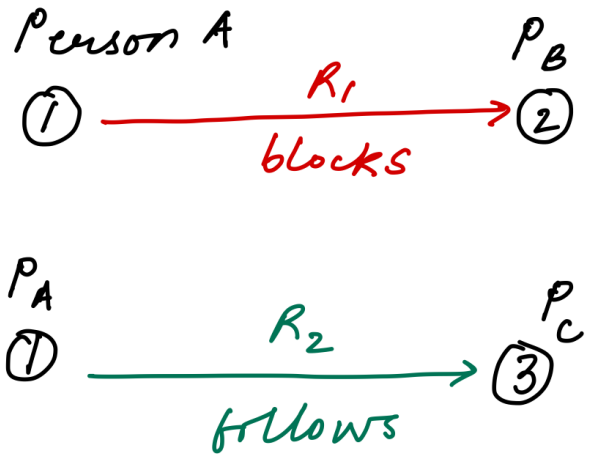
The GCN model requires homogenous graph. GCNs use a single weight matrix for each layer which cannot utilize the information from different types of edges.

Heterogenous graph can be converted to relational graph with the meta path concept.

The R-GCN uses separate weight matrices for separate relations in order to uniquely use different relations to update the node embeddings.

RELATIONAL GRAPH CONVOLUTIONAL NEURAL NET





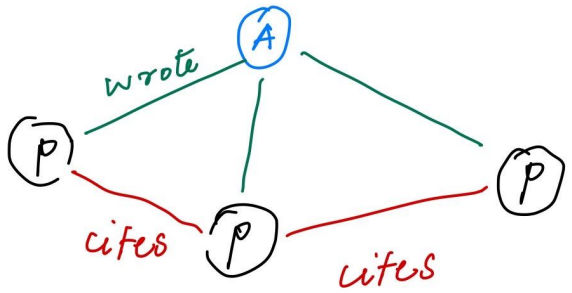
$$\sum_{j \in N_1} \frac{1}{c_{1j}} W_1 h_j$$

$$\sum_{j \in N_i} \frac{1}{c_j} W_2 h_j$$

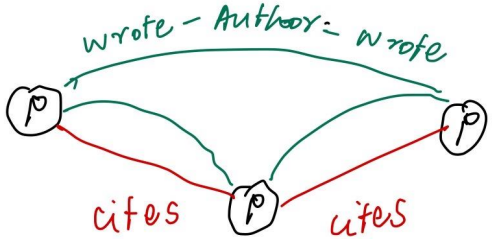
$$h_i' = \sigma\left(\sum_{r \in R} \sum_{j \in N_i} \frac{1}{c_{ij}} W_r h_j \right)$$

$$h_i' = \sigma\left(W_0 h_i + \sum_{r \in R} \sum_{j \in N_i} \frac{1}{c_{ij}} W_r h_j \right)$$

RELATIONAL GRAPH CONVOLUTIONAL NEURAL NET



Heterogeneous graph



Multirelational Graphs

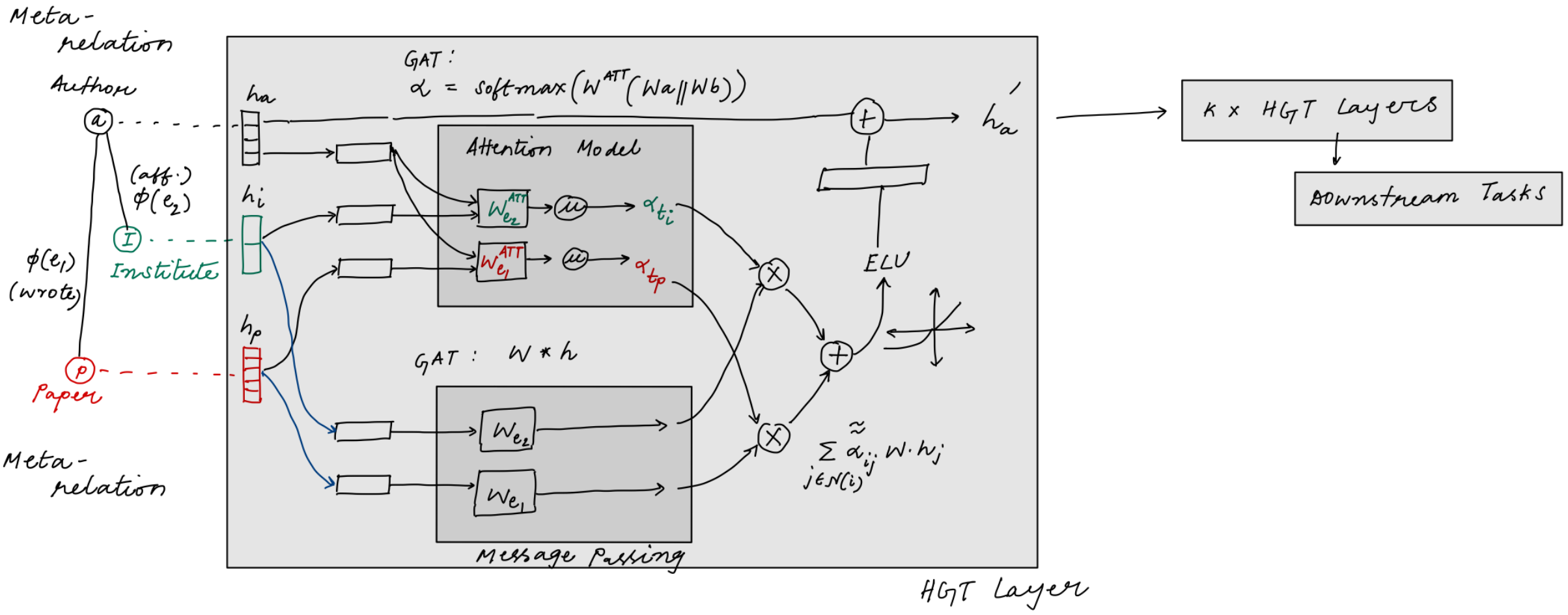
$$h_i^{l+1} = \sigma\left(W_0^l h_i^l + \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{ij}} W_r^l h_j^l\right)$$

> Metapath decomposition is done by GNN researcher who may not have domain expertise on the dataset
> Loss of information due to simplification.

Heterogeneous Graph Transform

> Introduce meta-relations
> Use GAT like computation.

HETEROGENOUS GRAPH TRANSFORMER



https://arxiv.org/pdf/2003.01332.pdf

LOSS FUNCTION DEPENDING ON TASK

- **Node Classification:**

$$\mathcal{L}(y_v, \hat{y_v}) = -\sum_c y_{vc} \log \hat{y_{vc}}.$$

where $\hat{y}_{vc}$ is the predicted probability that node v is in class c.

So, $-\sum_c y_{vc} \log \hat{y}_{vc}$ is the total loss of all the classes of a multiclass classification.

eg:

	TARGET	PREDICTION		$\mathcal{L}(y_{vc}, \hat{y_{vc}})$
Science	1	0.7	→	0.356
Math	0	0.2	→	0
English	0	0.1	→	0

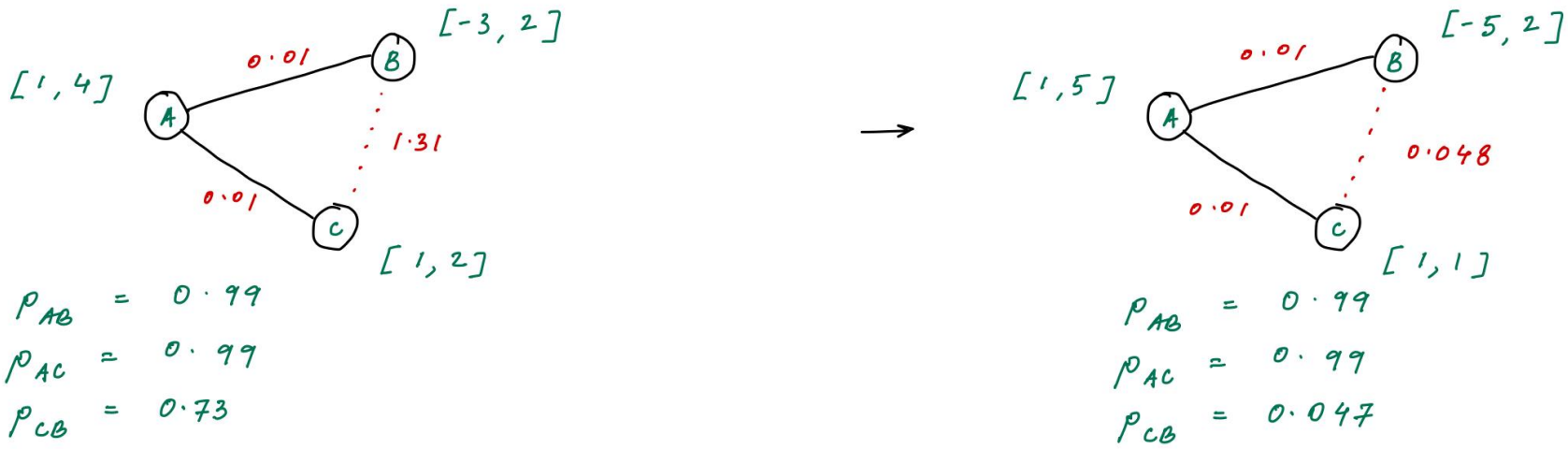
$\mathcal{L}(y_v, \hat{y_v}) = 0.356$

LOSS FUNCTION DEPENDING ON TASK

- **Link Prediction:** $\mathcal{L}(y_v, y_u, e_{vu}) = -e_{vu}\log(p_{vu}) - (1 - e_{vu})\log(1 - p_{vu})$

$$p_{vu} = \sigma(y_v^T y_u)$$

where e_{uv} is 1 when there is an edge between u and v node.
Let's see an example



CHALLENGES TO GRAPH NEURAL NET COMMUNITY

Dataset	Date	Node emb size	#nodes (millions)	#edges (millions)	#labeled nodes (millions)
OBGN-papers	2020	128*	111	1,615	1.4
MAG240M	2021	768	260	1,300	1.4
PinSAGE dataset	2018	128-2K (avg. 1K)	3000	18,000	UNK

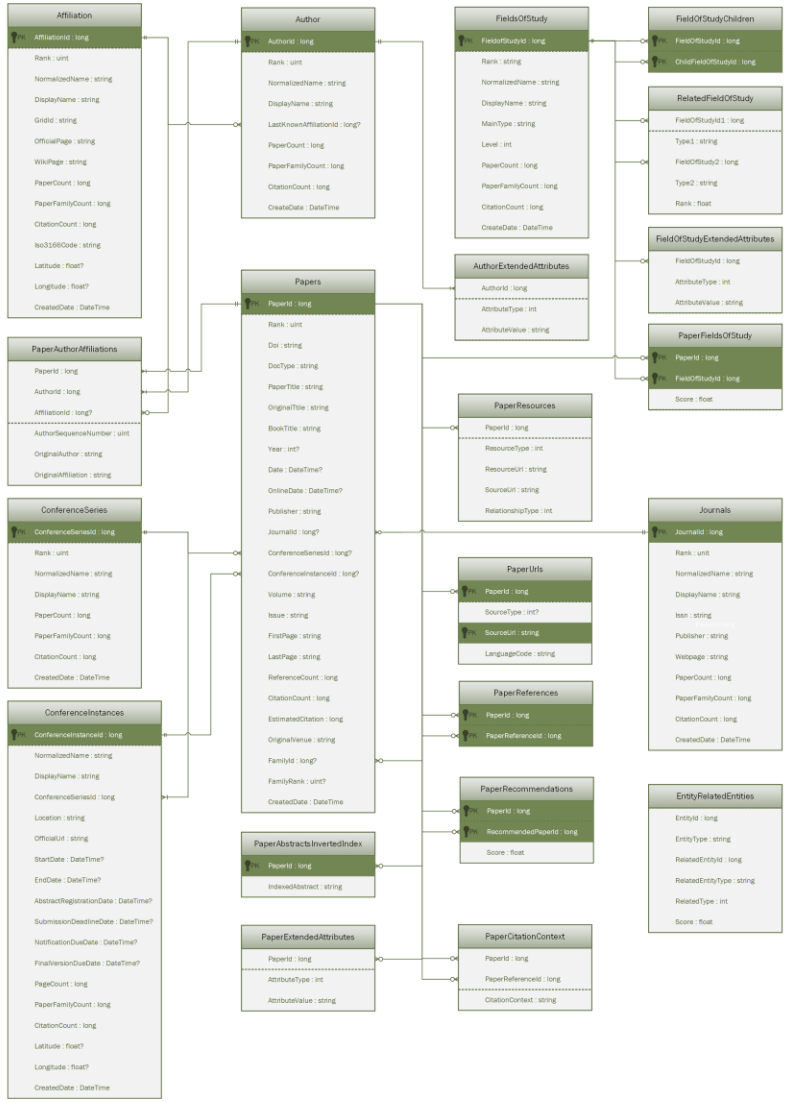
- Large gap between publicly available graph datasets and ones used by industry.
- Our goal is to propose a new dataset that will help both system designers and GNN researchers in two ways:

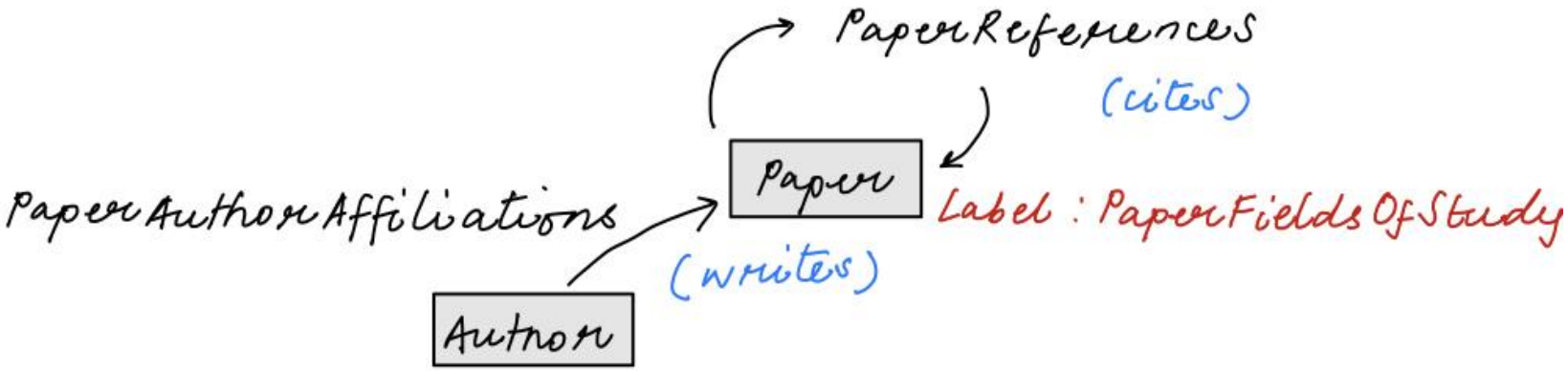
- Given a dataset schema, propose a methodology to generate arbitrary sized graphs (homogenous or heterogenous) and node embeddings with prescribed number of nodes, edges and relations.

- Provide a dataset with synthetic node embeddings for system developers and another dataset with node embeddings generated using NLP methods for GNN researchers and system developers.

* → paper does not provide information. Calculated via dataset size.

ILLINOIS GRAPH BENCHMARK (IGB)



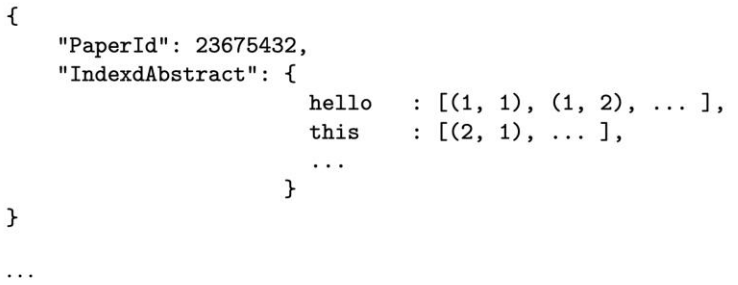


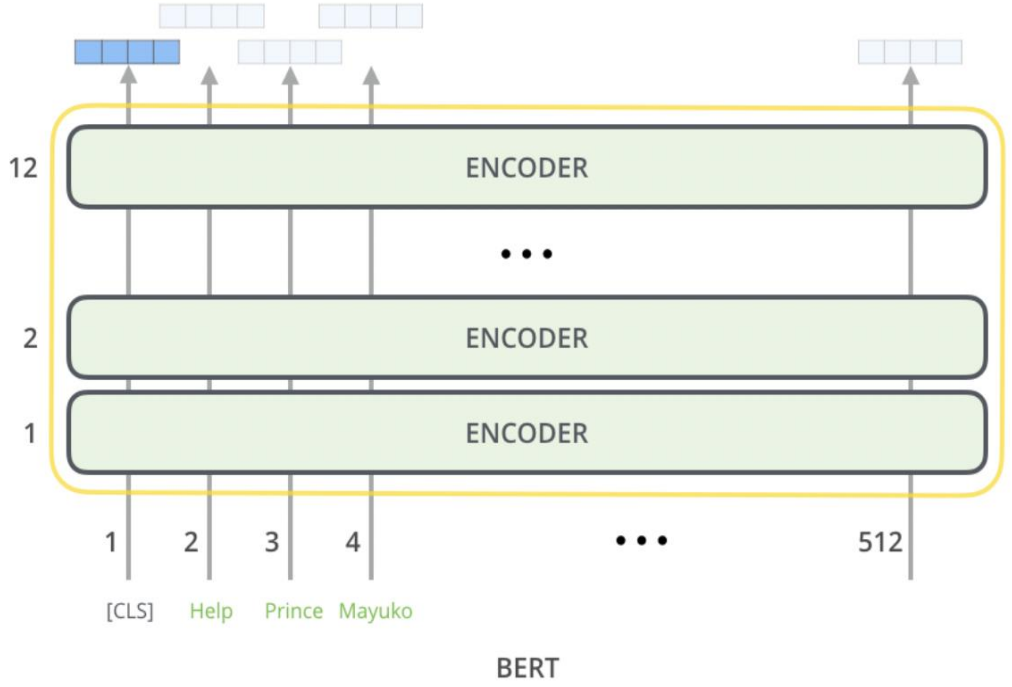
GENERATION OF NODE EMBEDDINGS

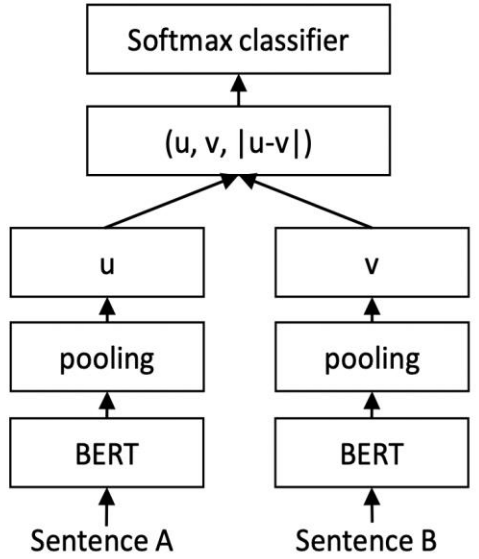
From the MAG dataset we extracted abstracts for each paper which is stored in an inverted index format.

After converting it into text and cleaning the data, we used the Sentence-BERT model to generate node embeddings for each paper node.

The BERT model takes the entire abstract and tokenizes each word using a layer of Encoders which is then pooled together to get a single embedding using the Sentence-BERT model.







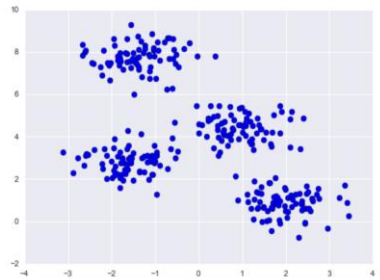
Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer/

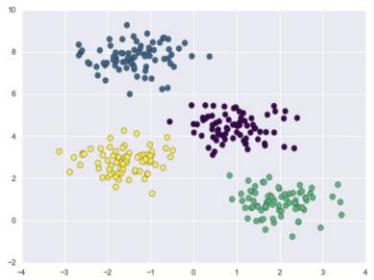
HOW WE CREATED THE LABELS

Challenges: The largest graph dataset that is available to the public today has only 1.4 million labeled nodes.

1. To over come this limitation in addition to the arxiv labels, we have incorporated an additional database namely the Semantic Scholar to get an additional 100 million labeled nodes out of 260 million nodes.


```
{
    "id": "4cd223df721b722b1c40689caa52932a41fcc223",
    "fieldsOfStudy": ["Computer Science"],
}
```



2. To generate even more labels for large prediction models we used an unsupervised K-Means algorithm using the journal and conferences information of each paper to cluster them into K classes. This works in two-folds: it allows the user to train on more data and allows the user to select the number of classes to get a finer or coarser classification.

CURRENT STATUS

Dataset	Date	Type	Node emb size	#nodes (millions)	#edges (millions)	#labeled nodes (millions)
OBGN-papers	2020	Real	128*	111	1,615	1.4
MAG240M	2021	Real	768	260	1,300	1.4
IGB v1	**2022**	**Real/Synthetic**	**128 - 4kB**	**267**	**1,900**	**~130+**
IGB v2	**2022**	**Real/Synthetic**	**variable**	**~600+**	**~ 3,000+**	**~200+**
PinSAGE dataset	2018	Real	128-2K	3000	18,000	NA

- Uses Sentence-BERT model to generate “real” node embeddings .
- Synthetic embedding vs real embedding:
  - Fundamentally GNN's find structural information of the graph to improve the embeddings and have no idea about the node's information.
  - Synthetic graph dataset would be useful to test computation and optimization the results
  - Synthetic graph datasets do not have any real-world significance for downstream tasks. For example, GAT would be completely useless for a synthetic dataset
- Then we will run GNN models to establish baseline results.

CONCLUSION

- Overall, our goal is to help the growth of the Graph Neural Net community
  - By giving GNN researchers access to larger labelled real graph datasets to test and develop more robust models.
  - By providing system researchers access to large synthetic datasets to improve the hardware and systems so we can process large amount of data more efficiently.

We are continuously running baseline experiments to confirm the working on the dataset and prove that larger datasets lead to higher baseline accuracy in supervised tasks.

If you have any questions about Graph Neural Nets or the Illinois Graph Benchmark at a later point, feel free to reach out to me at akhatua2@Illinois.edu

Thank you!

Please feel free to ask me any questions

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN